ECE 538 - Advanced Computer Architecture
Homework Assignment 2 (Individual)

100 points

Assigned Date: 10/11/2021 **Due Date:** 10/21/2021

David Kirby   davidkirby@unm.edu

1. **(20 points)** Answer the following qualitative questions:

   (a) (5 points) What are the types of cache misses discussed in lecture so far? Briefly describe each.

   1. Compulsory: These are the intitial misses due to an empty (cold) cache.
   2. Conflict: These are the misses due to a rigid block placement strategy (i.e. low associativity).
   3. Capacity: These are misses due to the cache being too small to hold the entire working set of data and instructions.

   (b) (5 points) How can pipelining the L1 cache improve processor performance? What negative effects can it have?

   Pipelining the L1 cache increases the cache bandwidth, but negatively affects the hit time.

   (c) (5 points) How does a direct-mapped cache compare to a set-associative cache in terms of access time, logic complexity, energy, etc.?

   Associativity reduces conflict misses by providing a more flexible block placement strategy. For example, direct-mapped has only one location in the cache that a block can be placed, but full-associativity allows blocks to be placed anywhere in the cache.

   While a set-associative cache decreases the miss rate, the drawbacks to associativity include increased complexity, access (hit) time, and power as well as added difficulty when pipelining the cache.

   (d) (5 points) What is Average Memory-Access Time (AMAT)? And how it can be calculated?

   Average memory-access time is an indirect measure of cache performance; although it is a better measure than miss rate, it is not a substitute for execution time.

$$\text{AMAT} = \text{hit time} + \text{miss rate} \times \text{miss penalty}$$

2. **(25 points)** Assume you have an *in-order* RISC processor that has a CPI of 1.3 with an ideal memory hierarchy (i.e., all memory accesses hit in the L1 caches). For a given benchmark, 38% of instructions executed are loads and stores. The miss penalty for reads and writes to the L1 data cache is 20 clock cycles and the miss rate is 3%.

   (a) (5 points) Calculate the CPI of your RISC processor taking memory stalls due to data accesses into account.

$$\begin{aligned} \text{CPI} &= \frac{\text{clock cycles}}{\text{instruction}} + \text{miss rate} \times \text{miss penalty} \\ &= 1.3 + 0.38 \times 0.03 \times 20 \\ &= 1.528\ \frac{\text{clock cycles}}{\text{instruction}} \end{aligned}$$

   (b) (5 points) The miss penalty for the L1 instruction cache is also 20 clock cycles and the miss rate is 1%. Recalculate the CPI of your processor taking into account stalls from both the instruction and data caches. How much faster is the machine with the ideal memory system?

$$\begin{aligned} \text{CPI} &= \frac{\text{clock cycles}}{\text{instruction}} + \text{miss rate} \times \text{miss penalty} \\ &= 1.3 + 0.01 \times 20 + 0.38 \times 0.03 \times 20 \\ &= 1.728\ \frac{\text{clock cycles}}{\text{instruction}} \end{aligned}$$

   (c) (5 points) Assuming your processor runs at 1 GHz, what is the average memory access time of the L1 instruction cache? What is the average memory access time of the L1 data cache?

$$1 \text{ GHz} = 1 \text{ ns/clock cycle}, \qquad \text{AMAT} = \text{hit time} + \text{miss rate} \times \text{miss penalty}$$

$$\begin{aligned} \text{AMAT}_I &= 1 \text{ ns} + 0.01 \times 20 \text{ ns} \\ &= 1.2 \text{ ns} \\ \text{AMAT}_D &= 1 \text{ ns} + 0.03 \times 20 \text{ ns} \\ &= 1.6 \text{ ns} \end{aligned}$$

   (d) (5 points) Why might the hit rate be higher for the instruction cache compared to the data cache?

Instruction cache exhibits a lot more spatial and temporal locality compared to data access cache.

(e) (5 points) Calculate the *misses per instruction* for both instruction and data caches. How does this metric differ from *miss rate*?

$$\frac{\text{Misses}}{\text{Instruction}} = \text{Miss rate} \times \frac{\text{Memory accesses}}{\text{Instruction}}$$

$$\begin{aligned}\frac{\text{Misses}}{\text{Instruction}}(I) &= 0.01 \times 1 \\ &= 0.01 \\ \frac{\text{Misses}}{\text{Instruction}}(D) &= 0.03 \times (1 + 0.38) \\ &= 0.0138\end{aligned}$$

This metric differs (as is shown in the equation above) by accounting for the memory accesses per instruction. In this example, 38% of our instructions access memory (in the form of data reads and writes); therefore, we see a higher miss per instruction.

3. **(30 points)** You are designing a System On a Chip (SoC) with your 5-stage pipelined RISC processor. The minimum system requirements are 512KB of program ROM and 16KB of RAM. For a given technology node, the critical path of your processor (ignoring memory access) is 1 ns. The access time for a 512KB ROM in the same technology is 2.0 ns, while the access time for a 16KB RAM is 0.8 ns.

   (a) (5 points) Assuming the Fetch and Memory stages of your processor only access memory and the flip-flop overhead is 0.1 ns, what is the fastest achievable clock rate of your processor when directly fetching instructions from the program ROM?

   (b) (5 points) Assuming accesses to program ROM are cached in a 4KB instruction cache with an access time of 0.5 ns, what is the fastest achievable clock rate of your processor?

   (c) (10 points) If the miss rate of the 4KB instruction cache with a 16 byte block size is 6.3% for a given workload, what is the average number of memory stall cycles per instruction your processor will see with a 4KB instruction cache? Assume the instruction cache can start a read from program ROM on the cycle after the miss is detected, and critical word first is not implemented.

   Hint: Start by calculating the miss penalty in clock cycles and remember a cache miss fills an entire line.

   (d) (10 points) Without the instruction cache, the processor is able to complete 4 million instructions from the same workload in 4.8 million clock cycles. What will be the CPI of your processor with the 4KB instruction cache, assuming the same workload?

4. **(25 points)** Consider the following simple C code for matrix transpose:

```
for (i = 0; i < 1024; i++){
    for (j = 0; j < 1024; j++){
        Y[j][i] = X[i][j]
    }
}
```

Assume that both X and Y are stored in the row major order and each element in the matrices is a double word (64-bit) integer.

(a) (5 points) Execute the code on your laptop and report the execution time. Would loop interchange help improve the performance of the code? Give explanation.

(b) (10 points) Rewrite the code by applying blocking (the software technique to improve data reuse). Use a blocking factor that leads to a noticeable performance improvement on the same computer as the one used in (a). Explain your reason for choosing the specific blocking factor.

(c) (10 points) Execute the new code and report the execution time. Make a comparison with the results reported in (a).